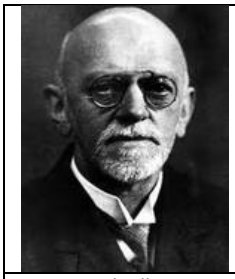
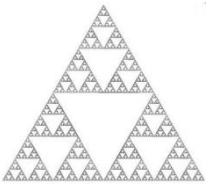
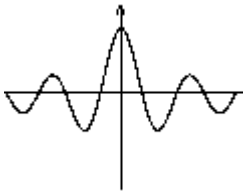
Objectifs :

- J'ai compris ce qu'est un algorithme.
- Je sais faire un tableau de déroulement pour tester un algorithme.
- Je sais repérer dans un algorithme la déclaration des variables, les entrées, le déroulement, les sorties.
- Je sais afficher un résultat, une question sur l'écran.
- Je sais récupérer une information, une valeur, un nom.
- Je sais déclarer une variable.
- Je sais affecter une valeur à une variable.
- Je sais reconnaître et utiliser la structure SI… ALORS… SINON
- Je sais reconnaître et utiliser la boucle POUR
- Je sais reconnaître et utiliser la boucle TANT QUE
- Je sais reconnaître, utiliser, concevoir des algorithmes utilisant plusieurs structures, qu'elles soient imbriquées, ou les unes à la suite des autres.

1°) D'où viennent les algorithmes ?

La notion d'algorithme vient de la volonté de trouver des règles écrites pour formaliser le calcul. L'idée remonte à l'Antiquité, mais, le concept prend un essor remarquable au XXe siècle. Tout commence avec l'ambition affirmée par David Hilbert (mathématicien allemand 1862-1943) de découvrir un algorithme universel. D'autres mathématiciens prouvent que c'est impossible. Il sera toutefois inévitable de recourir à une machine pour démontrer certains théorèmes. La définition abstraite d'une telle machine par Alain Turing (mathématicien, cryptologue et informaticien britannique, 1912-1954) et les besoins croissants en calcul font naître les premiers ordinateurs vers 1950. Ils favoriseront de nombreuses avancées : grâce à leurs capacités graphiques, Benoît Mandelbrot (mathématicien franco-américain, 1924-2010) découvre les fractales, des équipes françaises développent le traitement du signal par ondelettes, et le cinéma invente le film d'animation numérique.

				
David Hilbert.	*Alain Turing.*	*Benoît Mandelbrot.*	*Exemple de fractale.*	*Signal par ondelettes.*

Sources : les dossiers de la recherche, n°46, décembre 2011 ; Google Image

2°) A quoi ressemble un algorithme ?

A une recette de cuisine, à un mode d'emploi, à un exercice de mathématique, à une procédure… Lorsque l'on lit ou lorsque l'on rédige un algorithme, un certain nombre de données sont à prendre en compte :

Qui suis-je ? à qui je m'adresse ? dans quelle langue je vais parler ?

L'analyste programmeur doit « rentrer dans la peau de la machine » qu'il va programmer. Dans ce cas, à la question « qui suis-je ? », la réponse est : une calculatrice, un tableur…

A qui je m'adresse ? l'analyste programmeur, qui « est » la machine, s'adresse, le plus souvent, à un utilisateur.

Dans quelle langue je vais parler ? il faudra distinguer le langage de programmation (instructions que la machine doit réaliser) du langage de communication utilisé avec l'utilisateur (la machine peut, à l'aide de boîte de dialogue, d'affichages, ou de formulaires, échanger avec l'utilisateur).

Exemple : je souhaite écrire un algorithme permettant à la machine de demander le prénom à l'utilisateur, puis d'utiliser son prénom pour lui dire « bonjour ». Attention à ne pas confondre *lire* et *écrire* : on se met dans la peau de la machine…

LANGAGE NATUREL :	**ALGOBOX :**

Programme BONJOUR.
Variable : *nom* est une chaîne de caractères.
Début
  Ecrire « Comment tu t'appelles ? »
  Lire *nom*.
  Ecrire « Bonjour », *nom*.
Fin.
	1 VARIABLES
	2 nom EST_DU_TYPE CHAINE
	3 DEBUT_ALGORITHME
	4 AFFICHER* "Comment tu t'appelles?"
	5 LIRE nom
	6 AFFICHER "Bonjour "
	7 AFFICHER* nom
	8 FIN_ALGORITHME

Plusieurs remarques utiles :

- ✓ La syntaxe doit être très rigoureuse ! la moindre erreur de virgule ou autre ne permettra pas la réalisation du programme.
- ✓ On a utilisé, dans ces exemples, des variables : on avait besoin de *mémoriser une information*, il a donc fallu donner un nom à une case mémoire pour le langage naturel et le langage algobox. Lorsque l'on programme une calculatrice, on n'a pas besoin de déclarer les variables.
- ✓ Il existe de nombreux langage de programmation, plus vous êtes un analyste programmeur spécialisé, et moins votre langage de programmation est « naturel ». Les langages de programmations rencontrés dans le monde de l'informatique sont en anglais.

3°) Affectation de variable

Nous l'avons vu précédemment, un programme ou un algorithme peut avoir besoin de mémoriser une ou plusieurs valeurs (nombre, caractère, chaîne de caractère, booléen…).
A chaque valeur mémorisée correspondra une variable.
Dans les langages naturels, Algobox, et dans la grande majorité des langages de programmation, il faut déclarer ces variables et préciser quel est leur type (nombre, caractère, chaîne de caractère, booléen…).
Certaines calculatrices ne nécessitent pas la déclaration des variables.

Affecter une valeur à une variable, cela signifie mettre une valeur dans une variable.

Observons par exemple le programme suivant, qui demande une valeur à l'utilisateur, et lui renvoie son carré diminué de 1.

LANGAGE NATUREL

Programme CARRE1.

Variables :
X est un nombre.
Y est un nombre.

Début.
Ecrire « Donner X ».
Lire X.
X*X−1 → Y.
Ecrire Y.
Fin.

CALCULATRICE TI 83

```
PROGRAM:CARRE1
:Disp "DONNER X"
:Input X
:X*X-1→Y
:Disp "Y=",Y
```

ALGOBOX

```
VARIABLES
  X EST_DU_TYPE NOMBRE
  Y EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER* "Donner la valeur de X"
  LIRE X
  Y PREND_LA_VALEUR (X*X-1)
  AFFICHER* Y
FIN_ALGORITHME
```

CALCULATRICE CASIO GRAPH 35 SD

```
======CARRE1   ======
"X"↵
?→X↵
(X×X-1)→Y↵
"Y="↵
Y
```

Remarques :

- ✓ Pour s'assurer qu'un algorithme fonctionne correctement, il convient de le tester avec plusieurs valeurs et d'essayer de penser aux cas de figures qui pourraient être problématiques.
- ✓ La programmation sur calculatrice ne fonctionne pas forcément si vous n'avez pas pris les fonctions ou symboles au bon endroit. Aidez-vous de votre manuel d'instruction ! (téléchargez-les si vous les avez perdus)

4°) Le tableau de déroulement d'un algorithme

Observons à présent le programme suivant, qui demande deux valeurs à l'utilisateur, qui demande une valeur numérique pour la variable X, une valeur numérique pour la variable Y, et qui permute ces deux valeurs.
Nous aurons besoin d'une variable interne au programme (variable dont l'utilisateur ne connaît pas l'existence).

LANGAGE NATUREL

Programme PERMUTE.

Variables :
X est un nombre.
Y est un nombre.
Z est un nombre.

Début.
Ecrire : « Donner la valeur de X. ».
Lire X.
Ecrire : « Donner la valeur de Y. ».
Lire Y.
X → Z.
Y → X.
Z → Y.
Ecrire « X = », X.
Ecrire « Y = », Y.
Fin.

Le tableau de déroulement permet d'accomplir, pas à pas, ce que fait la machine qui exécute l'algorithme. Il faudra une colonne par variable, une ligne par instruction. En entrée, nous indiquerons les données récupérées par l'utilisateur. En sortie, celles affichées à l'écran. De ce fait, le tableau de déroulement ne commencera qu'à l'instruction X → Z et se finira à l'instruction Z → Y.

Pour faire un tableau de déroulement, il faut choisir avec quelle valeur nous allons tester l'algorithme. Dans le tableau suivant, nous testons l'algorithme avec X=8 et Y=2.

variables	X	Y	Z
entrées	8	2	
déroulement			8
	2		
		8	
sorties	2	8	

5°) Si… Alors… Sinon…

Dans le programme qui suit, on détermine si un triangle est rectangle. Il revient à l'utilisateur de donner les mesures des trois côtés. On nommera a le côté le plus long, b et c les deux autres côtés.

LANGAGE NATUREL

Programme TRIRECT.

Variables :
a est un nombre.
b est un nombre.
c est un nombre.

Début.
Ecrire « Donner la mesure du côté le plus long. »
Lire a.
Ecrire « Donner les deux autres mesures. ».
Lire b.
Lire c.
Si ($a^2=b^2+c^2$)
Alors
Ecrire « Le triangle est rectangle ».
Sinon
Ecrire « Le triangle n'est pas rectangle ».
Fin si.
Fin.

CALCULATRICE TI 83

Pour les programmes sur calculatrice, on supposera, pour simplifier, que l'utilisateur sait qu'il doit rentrer d'abord la mesure du côté le plus long, et ensuite les deux autres mesures.

```
PROGRAM:TRIRECT
:Input A
:Input B
:Input C
:If (A*A=B*B+C*C
)
:Then
:Disp "LE TRIANG
LE EST RECTANGLE
"
:Else
:Disp "LE TRIANG
LE N EST PAS REC
TANGLE"
:End
```

LANGAGE ALGOBOX

```
VARIABLES
  a EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
  c EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER* "Donner la mesure du côté le plus long."
  lire a
  AFFICHER* "Donner la mesure des autres côtés."
  LIRE b
  LIRE c
  SI ((a*a)==(b*b+c*c)) ALORS
      DEBUT_SI
        AFFICHER* "Le triangle est rectangle."
      FIN_SI
  SINON
      DEBUT_SINON
        AFFICHER* "Le triangle n'est pas rectangle."
      FIN_SINON
FIN_ALGORITHME
```

CALCULATRICE CASIO GRAPH 35 SD

Pour les programmes sur calculatrice, on supposera, pour simplifier, que l'utilisateur sait qu'il doit rentrer d'abord la mesure du côté le plus long, et ensuite les deux autres mesures.

```
======TRIRECT ======
"A="?→A↵
"B="?→B↵
"C="?→C↵
If (A×A=B×B+C×C)↵
Then "LE TRIANGLE EST
 RECTANGLE"↵
Else "LE TRIANGLE N'E
ST PAS RECTANGLE"↵
IfEnd
```

CASIO : vous trouverez les commandes en faisant Shift, Prgm, F1.
TI : vous pouvez passer par le catalogue ou par la touche PRGM.
Pour le tableau de déroulement, il faudra rajouter une colonne pour les conditions.

6°) Boucle POUR

L'algorithme qui suit permet de faire la somme de tous les nombres compris entre deux entiers a et b, entiers donnés par l'utilisateur.

LANGAGE NATUREL

Programme SOMMEAB

Variables

a est un nombre.
b est un nombre.
s est un nombre. *(la somme, calculée peu à peu)*
i est un nombre. *(variable locale, compteur de boucle)*

Début.
Ecrire « Donner le plus petit nombre. »
Lire a.
Ecrire « Donner le plus grand nombre. »
Lire b.
0 → s. *(initialisation)*
Pour i **de** a **à** b **faire** :
Début pour.
$s + i \rightarrow s$.
Fin pour.
Ecrire « la somme est : », s.
Fin.

LANGAGE TEXAS INSTRUMENT 83

```
PROGRAM:SOMMEAB
:Input A
:Input B
:0→S
:For(I,A,B)
:S+I→S
:End
:Disp S
```

Remarque : pour trouver les commandes diverses, vous pouvez utiliser le catalogue ou bien passer par PRGM.

LANGAGE ALGOBOX

```
1  VARIABLES
2    a EST_DU_TYPE NOMBRE
3    b EST_DU_TYPE NOMBRE
4    s EST_DU_TYPE NOMBRE
5    i EST_DU_TYPE NOMBRE
6  DEBUT_ALGORITHME
7    AFFICHER* "Donner la valeur du plus petit nombre."
8    LIRE a
9    AFFICHER* "Donner la valeur du plus grand nombre."
10   LIRE b
11   s PREND_LA_VALEUR 0
12   POUR i ALLANT_DE a A b
13       DEBUT_POUR
14         s PREND_LA_VALEUR s+i
15       FIN_POUR
16   AFFICHER "La somme est "
17   AFFICHER* s
18 FIN_ALGORITHME
```

LANGAGE CASIO GRAPH 35 SD

```
======SOMMEAB ======
?→A↵
?→B↵
0→S↵
For A→I To B↵
S+I→S↵
Next↵
S↵
```

Remarque : pour trouver les commandes de la boucle POUR, vous pouvez faire SHIFT, PRGM, F1, F6.

Pour le tableau de déroulement, il faudra rajouter une colonne pour la condition répétition de la boucle.
Dans certains algorithmes il est très utile de pouvoir ajouter des commentaires, afin d'expliquer le rôle d'une variable ou d'une manipulation, et de rendre l'algorithme plus facilement compréhensible. Chaque langage de programmation a sa propre façon d'écrire un commentaire.

7°) Boucle TANT QUE

L'algorithme qui suit génère un nombre secret compris entre 0 et 10, l'utilisateur doit réussir à le deviner en un minimum de coups. L'algorithme précisera en combien de coups l'utilisateur a trouvé le chiffre.

LANGAGE NATUREL

Programme DEVINE

Variables
secret est un nombre
tentative est un nombre
nbessai est un nombre

Début.
0 → *nbessai*
Nombre aléatoire entre 0 et 10 → *secret*
11 → *tentative*
Tant que *tentative* ≠ *secret* **faire**
Début tant que
Ecrire « tentez votre chance »
Lire *tentative*
nbessai + 1 → *nbessai*
Fin tant que.
Ecrire « Vous avez gagné en », *nbessai*, « coups. »
Fin.

LANGAGE TEXAS INSTRUMENT 83

Pour les programmes sur calculatrices, nous limiterons les noms de variables à une seule lettre.

```
PROGRAM:DEVINE
:0→N
:randInt(1,10)→S

:11→T
:While S≠T
:Input T
:N+1→N
:End
:Disp "GAGNE EN
"
:Disp N
:Disp "COUPS"
```

LANGAGE ALGOBOX

```
VARIABLES
  secret EST_DU_TYPE NOMBRE
  tentative EST_DU_TYPE NOMBRE
  nbessai EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  nbessai PREND_LA_VALEUR 0
  secret PREND_LA_VALEUR floor(random()*10+1)
  tentative PREND_LA_VALEUR 11
  TANT_QUE (tentative != secret) FAIRE
    DEBUT_TANT_QUE
      AFFICHER* "Tentez votre chance"
      LIRE tentative
      nbessai PREND_LA_VALEUR nbessai+1
    FIN_TANT_QUE
  AFFICHER "Vous avez gagné en "
  AFFICHER nbessai
  AFFICHER* " coups".
FIN_ALGORITHME
```

LANGAGE CASIO GRAPH 35 SD

```
======DEVINE  ======
0→N↵
Int (10×Ran# +1)→S↵
11→T↵
While T≠S↵
N+1→N↵
?→T↵
WhileEnd↵
"GAGNE EN"↵
N↵
"COUPS"
```

Remarque : on pourrait aussi faire en sorte que le nombre soit choisi par l'utilisateur.
Pour le tableau de déroulement, il faudra rajouter une colonne pour la condition de sortie de la boucle.